THE UNIVERSITY OF
WARWICK

Original citation:
Paterson, Michael S. and Zwick, U. (1990) Shallow multiplication circuits. University of Warwick. Department of Computer Science. (Department of Computer Science Research Report). (Unpublished) CS-RR-169

Permanent WRAP url:
http://wrap.warwick.ac.uk/60864

Copyright and reuse:
The Warwick Research Archive Portal (WRAP) makes this work by researchers of the University of Warwick available open access under the following conditions. Copyright © and all moral rights to the version of the paper presented here belong to the individual author(s) and/or other copyright owners. To the extent reasonable and practicable the material made available in WRAP has been checked for eligibility before being made available.

Copies of full items can be used for personal research or study, educational, or not-for-profit purposes without prior permission or charge. Provided that the authors, title and full bibliographic details are credited, a hyperlink and/or URL is given for the original metadata page and the content is not changed in any way.

A note on versions:
The version presented in WRAP is the published version or, version of record, and may be cited as it appears here.For more information, please contact the WRAP Team at:
publications@warwick.ac.uk

warwickpublicationswrap
highlight your research

http://wrap.warwick.ac.uk/

Research report 169

SHALLOW MULTIPLICATION CIRCUITS

MICHAEL S PATERSON, URI ZWICK

(RR169)

Ofman, Wallace and others used carry save adders to design multiplication circuits whose total delay is proportional to the logarithm of the length of the two numbers multiplied. An extension of their work is presented here.

The first part presents a general theory describing the optimal way in which given carry save adders can be combined into carry save networks.

In the second part, two new designs of basic carry save adders are described. Using these building blocks and the above general theory, the shallowest known theoretical circuits for multiplication are obtained.

Department of Computer Science
University of Warwick
Coventry CV4 7AL
United Kingdom

December 1990

Shallow Multiplication Circuits

Michael S. Paterson * *Uri Zwick* †

Abstract

Ofman, Wallace and others used carry save adders to design multiplication circuits whose total delay is proportional to the logarithm of the length of the two numbers multiplied. An extension of their work is presented here.

The first part presents a general theory describing the optimal way in which given carry save adders can be combined into carry save networks.

In the second part, two new designs of basic carry save adders are described. Using these building blocks and the above general theory, the shallowest known theoretical circuits for multiplication are obtained.

1. Introduction

This work examines theoretical ways to speed up multiplication circuits. The general approach used is the one suggested by Ofman [12] and Wallace [15]. The work presented here extends previous results reported in [13],[14].

The model used in this work is that of *dyadic Boolean circuits*. A dyadic Boolean circuit is a circuit composed of dyadic (i.e., two-input) Boolean gates. In some cases we consider circuits restricted to the *unate* gates, i.e., gates of the form $(x^a \wedge y^b)^c$, where $x^0 = x$ and $x^1 = \overline{x}$. These include the AND, OR, NAND and NOR gates, but exclude the XOR gate and its complement $(x \oplus y)^c$. The methods described in this work can also be used to construct fast multiplication circuits in cases where only a subset of the unate gates may be used. We allow the gates to be connected in an arbitrary acyclic manner and we assume that each one has a unit delay, i.e., the output of a gate is stabilized one unit of time after its two inputs are stabilized. The total delay of a circuit is the time that passes from the moment at which all the inputs are stable until all the outputs are stable. In this model the total delay corresponds to the length of the longest directed path from an input to an output.

This model ignores many practical considerations. No attention is paid for example to possible VLSI layouts of these circuits. Simplifying assumptions are made: that no delays are introduced on connecting wires and that the delay of a gate is not influenced by its surroundings, but this model enables a theoretical investigation of the inherent

*Department of Computer Science, University of Warwick, Coventry, CV4 7AL, England. This author was partially supported by the ESPRIT II BRA Programme of the EC under contract # 3075 (ALCOM).

†Mathematics Institute, University of Warwick, Coventry, CV4 7AL, England. This author was partially supported by the ESPRIT II BRA Programme of the EC under contract # 3075 (ALCOM).

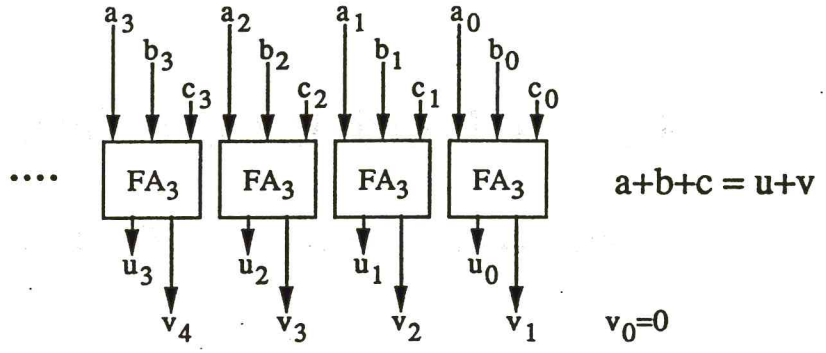

Figure 2.1. Constructing a $CSA_{3\to 2}$ using FA_3's.

delay needed in order to perform multiplication. Subsequent work may reveal ways of making the constructions described in this work more practical. The basic ideas of Ofman and Wallace, on which this work is based, are already of practical use (see e.g. [1]).

The above Boolean circuit model is the principal one used in theoretical investigations in the theory of computational complexity. For a summary of the extensive literature available on this subject the reader is referred to [4],[7],[16].

The first step in the Ofman-Wallace approach is the design of a *Carry Save Adder (CSA)*. The simplest *CSA* will receive three input numbers and avoid carry propagation by outputting the sum of them as the sum of two numbers. The striking discovery of Ofman, Wallace and others (see also [2],[5]) was that *CSA*'s can have constant delay independent of the size of the input numbers. A network of *CSA*'s is used to reduce the sum of n input numbers to the sum of only two. Such a network requires only a logarithmic number of layers and its total delay is therefore logarithmic. The two remaining numbers may be added using a *Carry Look Ahead* adder (see [3],[9]) which also has logarithmic delay.

Since multiplication of two n-bit numbers can be performed by adding n numbers the above construction yields logarithmic depth multiplication circuits.

2. Bit Adders and Carry Save Adders

The simplest $CSA_{3\to 2}$ is obtained by using an array of FA_3's (3-bit Full Adders) as shown in Fig. 2.1. In fact any *Bit Adder (BA)* could be used to construct a *CSA*.

A bit adder is a unit with k input bits and $\ell < k$ output bits. Each input and output bit has an associated significance. If the k input bits are denoted by $x_1, \ldots, x_k$ and their significances are $a_1, \ldots, a_k$, and if the ℓ output bits are denoted by $y_1, \ldots, y_\ell$ and their significances are $b_1, \ldots, b_\ell$ then the relation $\sum_{i=1}^{\ell} y_i 2^{b_i} = \sum_{j=1}^{k} x_j 2^{a_j}$ holds.

The simplest bit adder is the 3-bit full adder FA_3. The significance of the three input bits is 0 and the significances of the two outputs bits are 0 and 1 respectively. More generally, if $k = 2^m - 1$ then a unit FA_k which receives k input bits, and outputs m bits containing the binary representation of their sum could be constructed. The significance of all the inputs to an FA_k is again 0, and the significances of the m outputs are $0, 1, \ldots, m-1$.

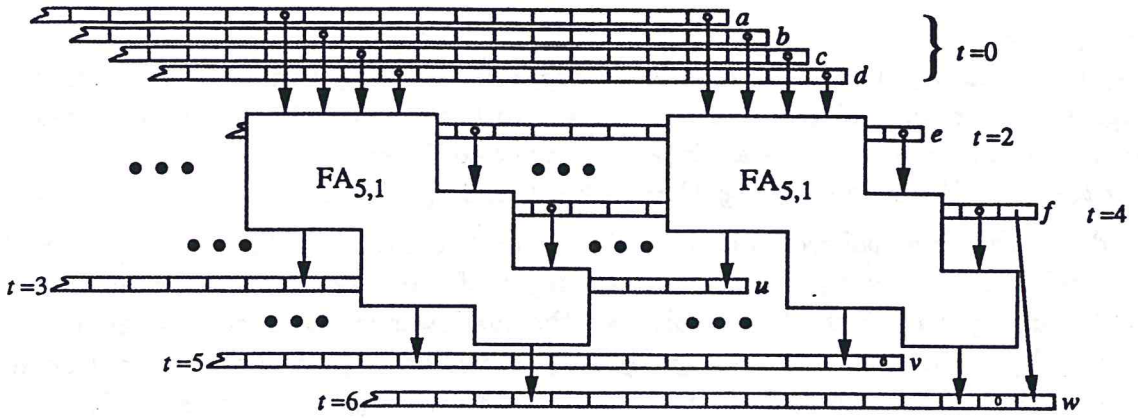

Figure 2.2. Constructing a $CSA_{6\to3}$ using $FA_{5,1}$'s.

The fastest *CSA* networks which can be constructed using $CSA_{3\to2}$'s have depths asymptotic to $3.71\log_2 n$ or $5.42\log_2 n$, depending on whether all dyadic gates or only the unate gates are used. In order to get our best performing *CSA*'s we need to look at slightly more general *BA*'s. If $\sum_{i=0}^{r} c_i 2^i = 2^m - 1$ we denote by $FA_{c_0,\ldots,c_r}$ the bit adder with $k = \sum_{i=0}^{r} c_i$ inputs, the first c_0 of them with significance 0, the next c_1 inputs with significance 1, and so on. The unit $FA_{c_0,\ldots,c_r}$ will have m outputs with significances $0, 1, \ldots, m-1$. Since every number $0 \le x < 2^m$ has a unique binary representation of length m, the output of this *BA* for any input vector is uniquely defined.

In section 4 we describe an efficient implementation of an $FA_{5,1}$. A $CSA_{6\to3}$ could be built using this $FA_{5,1}$ as illustrated in Fig. 2.2. The *CSA*'s constructed in this way give rise to the shallowest known multiplication circuits. These constructions use exclusive-or (XOR) gates. Their asymptotic delay is about $3.57\log_2 n$ time units.

In section 5 we describe an efficient implementation of an $FA_{7,4}$ using unate circuits. A $CSA_{11\to4}$ could be built using this $FA_{7,4}$ in a similar way to that shown in Fig. 2.2. The *CSA*'s constructed in this way yield the shallowest known multiplication circuits over the unate basis, and have an asymptotic delay of about $4.95\log_2 n$.

3. Generalised depth constructions

We consider networks composed of copies of some 'gadget' G which accepts k inputs at times $x_1, \ldots, x_k$, and delivers l outputs at times $y_1, \ldots, y_l$, where $k > l$. Some restrictions are imposed on $\vec{x}$ and $\vec{y}$ which reflect the functionality and minimality of the gadget G. The *characteristic polynomial* of G is $g(z) \equiv \sum_{j=1}^{l} z^{y_j} - \sum_{i=1}^{k} z^{x_i}$.

For any polynomial or power series $p(z)$, let $p(z)^{[<n]}$ denote the sum of those terms of p of degree less than n, and let $p(z)^{[n]}$ denote the remaining part.

In our intended applications, the gadget is a *CSA* so the inputs and outputs are nonnegative numbers and the sum of the inputs equals the sum of the outputs. A consequence of the functionality of the *CSA* is that, for any t, if all inputs at times less than t are zero, then any output at a time less than t is zero also. The device $G^{[t]}$, obtained by fixing to zero all inputs at times less than t and ignoring outputs at times less than t, is also a *CSA* with the correct functionality.

Let $t_{\min} = \min\{x_1, ..., x_k, y_1, ..., y_l\}$ and $t_{\max} = \max\{x_1, ..., x_k, y_1, ..., y_l\}$. Since any output at time $t_{\min}$ must be zero and any input at time $t_{\max}$ can be ignored, the lowest (highest) order term of $g(z)$ can be assumed to have a negative (positive) coefficient. Since input and output times are invariant under addition of a constant we may assume that $t_{\min} = 0$. Hence, $g(0) < 0$, $g(1) = l - k < 0$ and $g(\infty) = \infty$. Let $m = t_{\max} = \deg(g)$.

We define the *principal root* of G to be the smallest real root of $g(z)$ that is greater than one, and denote it by λ_G. The asymptotic depth of the networks we construct depends on the principal root; the larger this root the shallower is the circuit. Sometimes, for some t, $\lambda_{G[t]}$ is defined and $\lambda_{G[t]} \geq \lambda_G$, i.e., G can be improved by eliminating some initial inputs and outputs to it. If, for all $0 < t \leq m$, $\lambda_{G[t]}$ is undefined or $\lambda_{G[t]} < \lambda_G$, then G is said to be *reduced.*

For any polynomial or power series $f(z) = \sum_{i \geq 0} f_i z^i$ we define $f \cdot> 0$ ($f \cdot\geq 0$) if $f_i > 0$ ($f_i \geq 0$) for $0 \leq i \leq \deg(f)$ and also write, for example, $f \leq\cdot\, g$ when $(g - f) \cdot\geq 0$.

We define $\lfloor f \rfloor$ ($\lceil f \rceil$) to be the polynomial with coefficients $\lfloor f_i \rfloor$ ($\lceil f_i \rceil$).

Theorem 3.1 *If G is reduced then $g(z) = (z - \lambda_G)h(z)$ for some $h \cdot> 0$ with $\deg(h) = m - 1$.*

Proof : We have $h_{m-1} = g_m > 0$. Suppose that for some t, $0 \leq t < m - 1$, we have $h^{[t+1]} \cdot> 0$ but $h_t \leq 0$. Then

$$g^{[t+1]}(z) = h_t z^{t+1} + (z - \lambda_G)h^{[t+1]}(z) < 0$$

for $1 \leq z < \lambda_G$, and $g^{[t+1]}(\lambda_G) \leq 0$. Hence $\lambda_{G[t+1]} \geq \lambda_G$, which contradicts the assumption that G is reduced. □

We can interpret the equation $\lambda_G h(z) + g(z) = zh(z)$ as an assertion that if $\lambda_G h_i$ data items are available at time i, for $0 \leq i \leq m - 1$, and (some of them) are input to a copy of gadget G, then the result is that h_i items are available at time $i + 1$ for $1 \leq i \leq m$. Of course $\lambda_G h_i$ and h_i are in general nonintegral and at most $\lfloor \lambda_G h_i \rfloor$ inputs are used by G at time i, but it is convenient in our construction to associate these real numbers with each occurrence of G.

Let $b = 1/(\lambda_G - 1)$.

Lemma 3.2 *For all $A > 0$, $\lceil A + b \rceil < \lceil A/\lambda_G + b \rceil \lambda_G$.* □

Lemma 3.3 *For all integers $N > 0$,*

$$N + \sum_{t=0}^{K} \lceil A/\lambda_G^t + b \rceil z^t g(z) \leq\cdot\, z^{K+1}(b + 2)h(z),$$

where $A = \frac{N}{\lambda_G h(0)} - b$ and $K = \lceil \frac{\log A}{\log \lambda_G} \rceil = \lceil \frac{\log N}{\log \lambda_G} \rceil + O(1)$.

Proof :

$$N + \sum_{t=0}^{K} \lceil A/\lambda_G^t + b \rceil z^t g(z) \quad = \quad N - \lceil A + b \rceil \lambda_G h(z) +$$

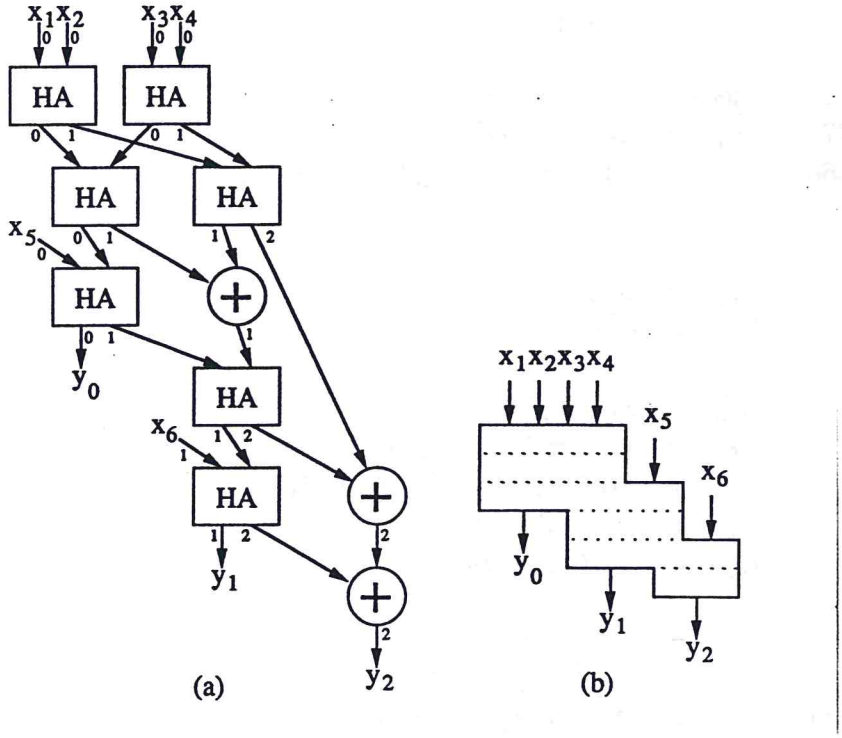

Figure 4.1. An implementation of an $FA_{5,1}$.

$$
\begin{aligned}
&+\lceil A/\lambda_G^K + b\rceil z^{K+1}h(z)\\
&+\sum_{t=1}^{K}\left(\lceil A/\lambda_G^{t-1} + b\rceil - \lceil A/\lambda_G^t + b\rceil \lambda_G\right) z^t h(z)\\
\leq\cdot\ & \lceil A/\lambda_G^K + b\rceil z^{K+1}h(z)\\
\leq\cdot\ & z^{K+1}(b+1)h(z).
\end{aligned}
$$

□

A copy of G which takes its first input at time t is said to be *based at time t* and has characteristic polynomial $z^t g(z)$. By the last lemma, if N inputs are supplied at time 0 to a network consisting of $\lceil A/\lambda_G^t + b\rceil$ copies of G based at time t for $t = 0, \ldots, \lceil (\log A)/\log \lambda_G\rceil$, then if dummy zero inputs are supplied wherever required the resulting network yields $O(1)$ outputs within a time which is $\log N/\log \lambda_G + O(1)$.

4. A $6 \rightarrow 3$ Carry Save Adder

An implementation of an $FA_{5,1}$ is given in Fig. 4.1(a). The implementation uses seven *Half Adders (HA)* and three XOR gates. The three XOR gates could in fact be replaced by OR gates. The left output of an *HA* with inputs a, b is $a \oplus b$ (the sum) and the right output is $a \wedge b$ (the carry). Thus an *HA* is composed of an XOR gate and an AND gate.

In order to verify the validity of this implementation we imagine at first that the three XOR gates are replaced by *HA*'s. Notice that the connections between the *HA*'s respect the significances of the inputs and outputs. The significance associated with each wire in the circuit is written next to that wire in Fig. 4.1(a). The inputs $x_1, \ldots, x_5$ have significance 0 while x_6 has significance 1. Finally, it is easy to check that the carry

Notation: $x = \underbrace{x_1x_2x_3}_{u}\underbrace{x_4x_5x_6x_7}_{v}$

$$
\begin{array}{lrcl}
4666666 & y_0 & = & \mathbf{S}_7^{1357} = U_{02}V_{13} \vee U_{13}V_{024} \\
5667777 & y_1 & = & \mathbf{S}_7^{2367} = U_{23}V_{04} \vee U_{12}V_1 \vee U_{01}V_2 \vee U_{03}V_3 \\
5666666 & y_2 & = & \mathbf{S}_7^{4567} = V_4 \vee U_{123}V_{34} \vee U_{23}V_{234} \vee U_3V_{1234}
\end{array}
$$

$$
\begin{array}{lrcl}
233 & U_{01} & = & \overline{U}_{23} \\
244 & U_{02} & = & \overline{U}_{13} \\
233 & U_{12} & = & \overline{U}_{03}
\end{array}
$$

$$
\begin{array}{lrcl}
122 & U_3 & = & x_1(x_2x_3) \\
233 & U_{03} & = & \overline{x}_1(\overline{x}_2\overline{x}_3) \vee x_1(x_2x_3) \\
244 & U_{13} & = & \overline{x}_1(\overline{x}_2x_3 \vee x_2\overline{x}_3) \vee x_1(\overline{x}_2\overline{x}_3 \vee x_2x_3) \\
233 & U_{23} & = & x_1(x_2 \vee x_3) \vee x_2x_3 \\
122 & U_{123} & = & x_1 \vee (x_2 \vee x_3)
\end{array}
$$

$$
\begin{array}{lrcl}
4444 & V_1 & = & \overline{x}_4\overline{x}_5(\overline{x}_6x_7 \vee x_6\overline{x}_7) \vee (\overline{x}_4x_5 \vee x_4\overline{x}_5)\overline{x}_6\overline{x}_7 \\
4444 & V_2 & = & V_{234}\overline{V}_{34} \\
4444 & V_3 & = & (\overline{x}_4x_5 \vee x_4\overline{x}_5)x_6x_7 \vee x_4x_5)(\overline{x}_6x_7 \vee x_6\overline{x}_7) \\
4444 & V_{13} & = & \overline{V}_{024}
\end{array}
$$

$$
\begin{array}{lrcl}
2222 & V_4 & = & x_4x_5x_6x_7 \\
3333 & V_{04} & = & \overline{x}_4\overline{x}_5\overline{x}_6\overline{x}_7 \vee x_4x_5x_6x_7 \\
3333 & V_{34} & = & (x_4x_5 \vee x_6x_7)(x_4x_6 \vee x_5x_7) \\
4444 & V_{024} & = & (\overline{x}_4\overline{x}_5 \vee x_4x_5)(\overline{x}_6\overline{x}_7 \vee x_6x_7) \vee (\overline{x}_4x_5 \vee x_4\overline{x}_5)(\overline{x}_6x_7 \vee x_6\overline{x}_7) \\
3333 & V_{234} & = & (x_4 \vee x_5)(x_6 \vee x_7) \vee (x_4 \vee x_6)(x_5 \vee x_7) \\
2222 & V_{1234} & = & x_4 \vee x_5 \vee x_6 \vee x_7
\end{array}
$$

Figure 4.2. Khrapchenko's construction of an FA_7.

output of the three *HA*'s that we used to replace the XOR gates are always zero so the *HA*'s could be replaced by XOR gates or by OR gates.

Note that the input x_5 is supplied to this unit two units of time after $x_1, \ldots, x_4$ are supplied, that y_0 is then obtained one unit of time later even before x_6 need to be supplied. The outputs y_1 and y_2 are obtained one and two units of time after x_6 is supplied. This behaviour is depicted in Fig. 4.1(b). The $CSA_{6\to3}$ constructed using this $FA_{5,1}$ will have the same delay characteristics.

The results of the previous section give us the optimal way of combining these $CSA_{6\to3}$'s into networks. The delay of these networks for the (carry save) addition of n numbers will be approximately $\log_\lambda n \simeq 3.57 \log_2 n$ time units where $\lambda \simeq 1.21486$ is the root of the polynomial equation $\lambda^6 + \lambda^5 - \lambda^4 + \lambda^3 - \lambda^2 - 4 = 0$.

5. An $11 \to 4$ Carry Save Adder

The $CSA_{6\to3}$ described in the previous section relied heavily on the use of XOR gates. An XOR gate could always be replaced by three unate gates with a total delay of two

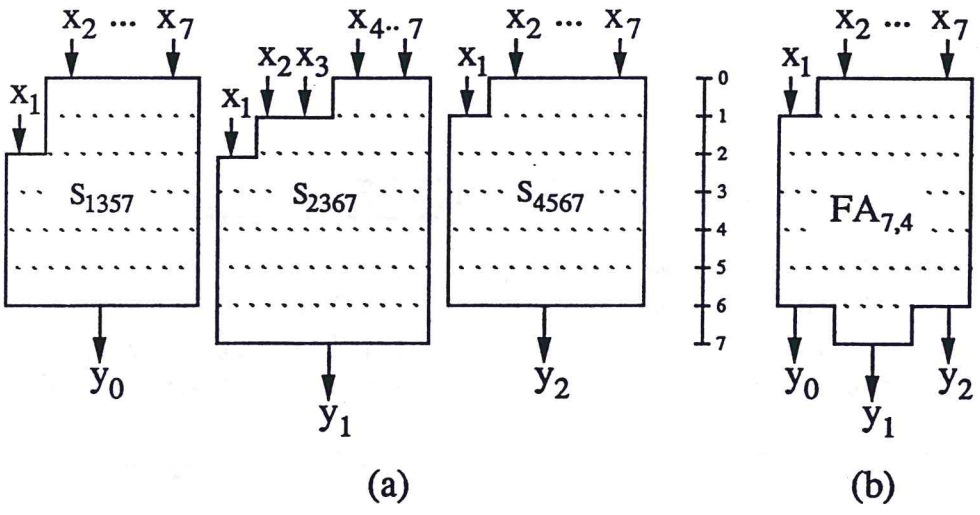

Figure 4.3. The delay characteristics of Khrapchenko's construction.

time units. Better results are obtained however by using a completely different design.

Khrapchenko [10] gave a design of an FA_7 using which a $CSA_{7\to 3}$ with the characteristics given in Fig. 5.2 could be constructed. He also described networks based on this *CSA* with asymptotic delay of $5.12\log_2 n$. The networks that he described were not optimal however. Using the designs of section 3, or even the less general designs described in [13],[14], better networks of delay $5.07\log_2 n$ can be obtained.

In this section we give an implementation of an $FA_{7,4}$ using which the preceding results can be further improved. Since the design of this unit is based on Khrapchenko's design, we give a concise summary of his construction in Fig. 5.1.

In Figs. 5.1 and 5.4 we use the following notation. We denote by $\mathbf{S}_k^A$ the symmetric function of k variables which takes the value 1 for inputs $x_1,\ldots,x_k$ if and only if $\sum x_i \in A$. For example, $\mathbf{S}_7^{4567}$ stands for the majority function on seven variables. For conciseness we write U_A for $S_3^A(u)$ where $u=(x_1,x_2,x_3)$, and V_A for $S_4^A(v)$ where $v=(x_4,x_5,x_6,x_7)$, and so on.

The numbers given to the left of each formula in Fig. 5.1 are the depths of the variables in that formula. They will be called *delay vectors*. The delay characteristics of the three output bits y_0,y_1,y_2 that compose Khrapchenko's FA_7 are described in Fig. 5.2(a). In [14] it is shown that the optimal way of combining these three units into a single unit is shown in Fig. 5.2(b).

The construction of the new $FA_{7,4}$ is given in Fig. 5.3. Figures 5.4 and 5.5 depict the final stages in the construction of y_3. The delay vectors of y_0,y_1,y_2,y_3 are shown in Fig. 5.6(a) and we see that we can fit them all into the unit outlined in Fig. 5.6(b).

By the results of section 3, we can combine the new $CSA_{11\to 4}$'s into networks of asymptotic depth $\log_\lambda n \simeq 4.95\log_2 n$ where $\lambda \simeq 1.15041$ is the positive root of the equation $2\lambda^9+\lambda^8+\lambda^6-4\lambda^2-\lambda-6=0$.

Notation: $x = \overbrace{\underbrace{x_1x_2x_3}_{u}\underbrace{x_4x_5x_6x_7}_{v}}^{s}\underbrace{x_8x_9x_{10}x_{11}}_{t}$

4666666	y_0	=	S_{1357}
78899996666	y_1	=	$S_{0145}T_{13} \vee S_{2367}T_{024}$
89999997777	y_2	=	$(S_{4567}T_{04} \vee S_{2345}T_1) \vee (S_{0123}T_2 \vee S_{0167}T_3)$
78888886666	y_3	=	$(S_{234567}T_{34} \vee S_{67}T_{1234}) \vee T_{234}(S_{4567} \vee T_4)$
56666664444	$S_{4567} \vee T_4$	=	$(U_{23}V_{234} \vee U_{123}V_{34}) \vee ((U_3V_{1234} \vee V_4) \vee T_4)$
5667777	S_{0145}	=	$\overline{S}_{2367}$
5666666	S_{0167}	=	$\overline{S}_{2345}$
5666666	S_{0123}	=	$\overline{S}_{4567}$
5666666	S_{2345}	=	$S_{234567}\overline{S}_{67}$
4555555	S_{234567}	=	$U_{123}V_{1234} \vee (U_{23} \vee V_{234})$
4555555	S_{67}	=	$U_{23}V_4 \vee U_3V_{34}$

Figure 5.1. The new $FA_{7,4}$ construction.

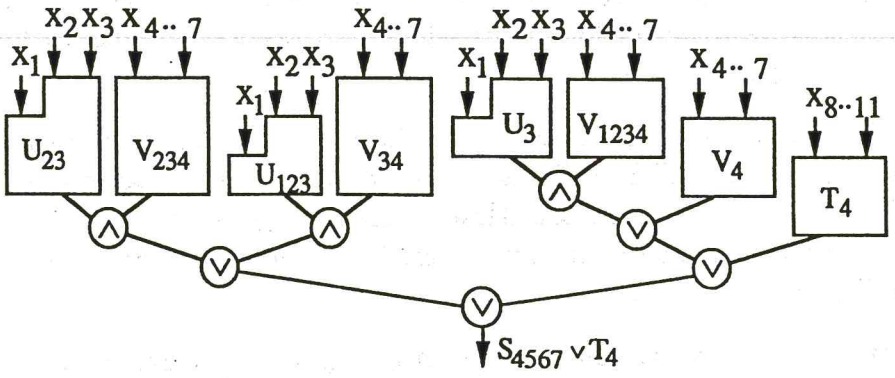

Figure 5.2. The final stages in the computation of $S_{4567} \vee T_4$.

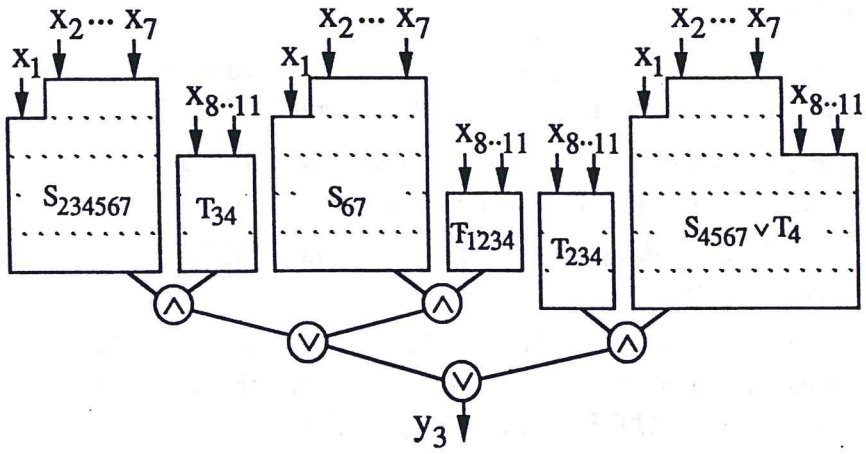

Figure 5.3. The final stages in the computation of y_3.

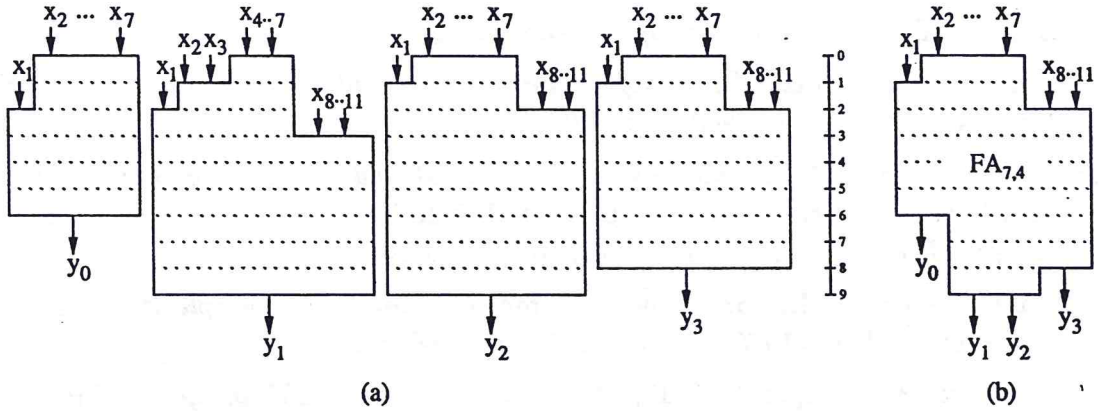

Figure 5.4. The delay characteristics of the new $FA_{7,4}$.

6. Concluding remarks

We have presented a general construction and some specific designs which yield circuits for carry save addition which are faster than those previously published. Although we have only given asymptotic results here, the same methods provide efficient networks for small numbers of inputs. There is a polynomial time algorithm which, for any *CSA* G and any n, gives an optimal-depth network of G's for the carry save addition of n inputs.

Our constants will no doubt be improved before long, but the techniques provide a simple construction method which may be of more durable value.

References

[1] El Gamal A., Gluss D., Ang P-H., Greene J., Reyneri J., *A CMOS 32 bit Wallace tree multiplier-accumulator. 1986 ISSCC Digest of Technical Papers, pp. 194-195.*

[2] Avizienis A., *Signed-digit number representation for fast parallel arithmetic. IEEE Trans. Elec. Comp. Vol. EC10 (1961), pp. 389-400.*

[3] Brent R., *On the addition of binary numbers. IEEE Trans. on Comp., Vol. C-19 (1970), pp. 758-759.*

[4] Boppana R., Sipser M., *The complexity of finite functions.* In *Handbook of Theoretical Computer Science Vol. A: Algorithms and Complexity*, ed. van Leeuwen, Elsevier/MIT Press, 1990, pp. 757-804.

[5] Dadda L., *Some schemes for parallel multipliers. Alta Frequenza, Vol. 34 (1965), pp. 343-356.*

[6] Dadda L., *On parallel digital multipliers. Alta Frequenza, Vol. 45 (1976), pp. 574-580.*

[7] Dunne P.E., *The complexity of Boolean networks. Academic Press, 1988.*

[8] Karatsuba A., Ofman Y., *Multiplication of multidigit numbers on automata. Soviet Physics Dokl., Vol. 7 (1963), pp. 595-596.*

[9] Khrapchenko V.M., *Asymptotic estimation of addition time of a parallel adder. Problemy Kibernet., Vol. 19 (1967), pp. 107-122* (in Russian). English translation in *Syst. Theory Res., Vol. 19 (1970), pp. 105-122.*

[10] Khrapchenko V.M., *Some bounds for the time of multiplication. Problemy Kibernet., Vol. 33 (1978), pp. 221-227* (in Russian).

[11] Mehlhorn K., Preparata F.P., *Area-time optimal VLSI integer multiplier with minimum computation time. Information and Control, Vol. 58 (1983), pp. 137-156.*

[12] Ofman Y., *On the algorithmic complexity of discrete functions. Doklady Akademii Nauk SSSR, 145 pp. 48-51* (in Russian). English translation in *Sov. Phys. Doklady, Vol. 7 (1963) pp. 589-591.*

[13] Paterson M.S., Pippenger N., Zwick U., *Faster circuits and shorter formulae for multiple addition, multiplication and symmetric Boolean functions. Proceedings of the 31st Ann. IEEE Symp. on Found. of Comp. Sci., St. Louis 1990.*

[14] Paterson M.S., Pippenger N., Zwick U., *Optimal carry save networks. Boolean function complexity: Selected papers from the LMS symposium, Durham 1990.* To appear, Cambridge Univ. Press, 1991.

[15] Wallace C.S., *A suggestion for a fast multiplier. IEEE Trans. Electronic Comp. EC-13 (1964) pp. 14-17.*

[16] Wegener I., *The complexity of Boolean functions. Wiley-Teubner Series in Computer Science, 1987.*